SYNTAX ZX80®

A PUBLICATION OF **THE HARVARD GROUP**

VOL.2 NO.5 ISSN 0273-2696 MAY, 1981

8K ROMS, NEW MACHINE FROM MICROACE

MicroAce will sell 8K ROMs for MicroAce and ZX8Ø computers within 8 weeks, according to Chris Cary, head of Compshop, Ltd. The ROMs are identical to Sinclair's and are licensed by them. You can order now, but MicroAce will not cash checks or process credit card numbers until they ship. The new ROM is $35.

By August MicroAce expects to ship their new MicroAce II. Also in kit form, the MicroAce II will have 16K RAM, 8K ROM, and a large keyboard. The price is not definite.

But MicroAce will not forget their MicroAce I customers. They are working on inexpensive add-on board to upgrade the original MicroAce to MicroAce II levels. This price is also indefinite. Cary said MicroAce is considering providing low-priced (about $25) big keyboards if there's enough interest.

MicroAce has no plans to make peripherals, such as printers. However, Cary said, MicroAce is interested in supporting outside producers who wish to make inexpensive add-ons.

Andy Fisher, general manager, is no longer with the company. Cary arrived from England to manage things until a new manager is chosen. According to Cary, machines sent in for repair are being shipped out the same day.

Contact MicroAce at:
1348 East Edinger
Santa Ana, CA 927Ø5
714/547-2526

SINCLAIR NEWS—ROMS AND RAMS HERE

By the time you read this, Sinclair will have delivered all 8K ROMs and 16K RAMs on order, according to Nigel Searle of Sinclair. A new, larger manual (over 200 pages) will accompany the ROMs. After this shipment, Sinclair US will be out of stock; expect more in about 1 month.

Nigel Searle will speak at the monthly meeting of the Philadelphia Area Computer Society, May 16, 2:00 PM, LaSalle College, on the ZX90 computer. New York readers can catch him in the last segment of WNYC's home electronics series, either May 8 or May 16.

NEW PUBLICATIONS

QZX newsletter serves amateur radio operators who own ZX8Øs. According to Marty Irons, K2MI, publisher, "It is a labor of love...I am funding it out of my own pocket, to provide a medium for amateurs to exchange programs, technical information and schematics to help them better utilize their ZX8Øs for amateur radio applications." No charge. Contact QZX, Marty Irons, 46 Magic Circle Dr., Goshen, NY, 1Ø924.

Micro Moonlighter newsletter will help you turn your microcomputer into a money machine. Marketing, advertising, contracts and industry trends will be covered in regular departments. The first issue will appear in May 81. Price: 12 issues $25, $29 in Canada, $35 worldwide. Contact J. Norman Goode, Publisher, 2115 Bernard Ave, Nashville, TN, 37212.

CHECK YOUR VOLTAGE

If you're having trouble loading your cassette on your MicroAce and have an oscilloscope, check the voltage on R34. For the most reliable loading, you should get a peak to peak voltage of 3.5-4.Ø volts. If the voltage is low, first try adjusting the volume. But if your tape or the automatic gain control on your recorder is bad, the amplitude will wander. If it drops below 2.75 V, the computer may miss a bit (binary digit). To fix this, short out R34 and change C13 to a 0.2 uF capacitor. This increases the voltage into U11 and compensates for low recorder output. Hint from Andy Fisher, formerly of MicroAce.

AUTOMATIC RECORDER CONTROL

You can call the LOAD routine directly from machine language by transferring control to Ø2ØØH. Then, if the tape recorder is running, the next program loads. To try this from BASIC, enter this one-line program:

```
1Ø LET U=USR(512)
```

Start the recorder. When the quiet leader comes up, run the program. In a machine language routine, call Ø2ØØH. Combine this with "Interface to the Real World" (Jan. 81) to control your recorder and automatically run program sequences. (Note: on our 1K ZX80, 518 as the decimal address worked; 512 did not.—AZ)

Tony Bringhurst, Cheektowaga, NY

SYNTAX ERROR: Joe Chaiet's Bar Chart program (Apr.81) contained an error. Line 134 should read:

```
IF Y<1Ø AND T>1 THEN PRINT ...
```

The Syntactic Sum is correct.

UPCOMING SHOWS AND EXHIBITIONS

May 2—Computer Fair for Small Business, Field House, North Campus, Middlesex Community College, Springs RD, Bedford, MA, Ø173Ø

May 21-23—TRS-8Ø Microcomputer Show, Statler Hilton Exhibition Hall, 7th Ave & 33rd St, New York

May 29-31—MACC, Franklin University, Grant & Main St, Columbus, OH, 432Ø2

May 31-June 3—Consumer Electronics Show, McCormick Place, Chicago, IL, 6Ø616

OUR POLICY ON CONTRIBUTED MATERIAL

SYNTAX ZX80 invites you to express opinions related to the ZX80 and the newsletter. We will print, as space allows, letters discussing items of general interest. Of course, we reserve the right to edit letters to a suitable length and to refuse publication of any material.

We welcome program listings for all levels of expertise. Programs can be for any fun or useful purpose. We will test run each one before publishing it, but we will not debug programs; please send only workable listings.

In return for your listing, we will pay you a token fee of $2.00 per program we use. This payment gives us the nonexclusive right to use that program in any form, world-wide. This means you can still use it, sell it, or give it away, and so can we.

We will consider submissions of news and hardware or software reviews. Please keep articles short (350–400 words). Again, we reserve the right to edit accepted articles to a suitable length. We will pay 7 cents per 6 characters, including spaces and punctuation, for accepted articles.

When you send in programs for possible publication in SYNTAX, please include the following information:

- How to operate the program, including what to input if it does not contain prompts.
- Whether you can run the program over again and how.
- How to exit the program.
- The Syntactic Sum (using the Syntactic Sum program in the February, 1981, issue).
- Whether it fits in 1K or 2K RAM (or 16K when available).

We pay for this explanatory text at the same rate as for articles in addition to payment for the program itself.

If you want us to return your original program listing or article, please include a self-addressed, stamped envelope. Otherwise, we cannot return submitted material.

SYNTAX ZX80 is published monthly by Syntax ZX80, Inc., a wholly-owned subsidiary of The Harvard Group, RD 2, Box 457, Bolton Road, Harvard, MA 01451. Telephone 617 / 456-3661. Subscriptions: 12 issues, $25. Single issues, $4.

Publisher: Kirtland H. Olson
Editor: Ann L. Zevnik
Associate Editor: Susan G. Barber
Technical consultant: Eric K. Olson

Printed by Joseph E. Marhefka, Jr.,
Clinton Offset Printers
Clinton, MA 01510.

 ISSN 0273-2696

HANGMAN

This program is a ZX80 version of the 2 person spelling game, Hangman. To use, type in the program then hit RUN (NL). While the second player looks away, type in a mystery word or expression and hit (NL). The second player sees a series of blanks, each representing one character. Guess a letter by pressing that key and then (NL). If you're right, ZX80 fills in all blanks for that letter. If you're wrong, it displays your incorrect guess and adds another part to the man. The game continues until you fill all the blanks, or you hang the man. Maximum word or phrase length is 14 characters. You don't have to guess spaces. Graphics: Line 80 shift Q; Line 301 5 shift W shift D; Line 400 5 spaces shift Q: Line 412 shift A; Line 416 5 spaces shift A; Line 418 4 spaces X 1 space. (Note: I put in a 36 character phrase and it worked fine.--AZ)

Richard Van Workum, Hanford, CA

```
  1Ø PRINT "WORD?"
  15 INPUT A$
  4Ø LET Z=Ø
  5Ø DIM T(25)
  55 DIM D(11)
  6Ø FOR X=Ø TO 25
  65 LET T(X)=22Ø
  7Ø NEXT X
  8Ø LET G$="■"
 1ØØ CLS
 1Ø4 LET A=Ø
 1Ø5 LET Q=Ø
 1Ø6 LET N=Ø
 11Ø LET T$=A$
 115 LET X=CODE(T$)
 12Ø LET N=N+1
 122 IF NOT CODE(T$)=CODE(G$) TH
EN LET A=A+1
 125 IF X<3Ø OR X>63 THEN GO TO
17Ø
 13Ø PRINT CHR$(T(X-38));
 135 IF T(X-38)=220 THEN LET Q=Q
+1
 14Ø LET T$=TL$(T$)
 16Ø GO TO 115
 17Ø IF X=1 THEN GO TO 19Ø
 175 PRINT CHR$(X);
 18Ø GO TO 14Ø
 19Ø PRINT
 191 IF Q=Ø THEN GO TO 52Ø
 195 IF A=N THEN GO SUB 25Ø
 196 GO TO 3Ø1
 2ØØ PRINT "LETTER?"
 21Ø INPUT G$
 22Ø LET T(CODE(G$)-38)=CODE(G$)
 23Ø GO TO 1ØØ
 25Ø LET Z=Z+1
 251 LET D(Z-1)=CODE(G$)
 252 RETURN
 3Ø1 PRINT "▬▬▬▬▬ "
 3Ø3 LET R=Ø
 3Ø4 FOR K=1 TO Z
 3Ø5 GO SUB 4ØØ+R
 3Ø6 LET R=R+2
 3Ø7 NEXT K
 3Ø8 PRINT
 31Ø FOR L=1 TO Z
 311 PRINT CHR$(D(L));
 312 NEXT L
 313 PRINT
 315 GO TO 2ØØ
 4ØØ PRINT "     ▮"
 4Ø1 RETURN
 4Ø2 PRINT "     )";
 4Ø3 RETURN
 4Ø4 PRINT "O";
 4Ø5 RETURN
 4Ø6 PRINT "("
 4Ø7 RETURN
 4Ø8 PRINT "     +"
 4Ø9 RETURN
 41Ø PRINT "    V";
 411 RETURN
 412 PRINT "▒";
 413 RETURN
 414 PRINT "V"
 415 RETURN
 416 PRINT "     ▒"
 417 RETURN
 418 PRINT "    X ";
 419 RETURN
 42Ø PRINT "X"
 421 PRINT
 5ØØ PRINT A$
 5Ø1 STOP
 52Ø PRINT "WINNER"
 521 STOP
SYNTACTIC SUM= 98Ø
```

FREQUENCY HISTOGRAM

This program tests how random the ZX8Ø/MicroAce's random number generator really is. It counts how often each number is produced by the random number generator. It also demonstrates the technique of scaling graphs.

N is the number of cells, or bars, in the histogram. L is the number of times the program loops and calls the random number generator. Line 2Ø allows you up to 2Ø cells. Line 4Ø produces a random number from 1 to N. Line 5Ø increments the corresponding cell. Thus if the random number 8 is returned 6 times during the program run, A(8)=6.

Lines 62-66 scale the stored data to make sure it fits on the screen. The program allows 24 characters for each line of the graph. Line 75 makes line numbers 1-9 space correctly. The comma at the end of line 8Ø ensures that each bar starts in the same place.

This program contains 1 deliberate bug. Try running it with N=2Ø, L=3ØØØ. The display file fills up the space required after the program end for the BASIC system to store intermediate variables.

Use this routine as a subroutine to plot data generated by another program with these changes:
Delete lines 1Ø-17, 3Ø, 4Ø, 6Ø.
Change lines 6Ø & 16Ø to RETURN.
Determine a value for J from the main program.
Call GO SUB 5Ø to store.
Fix the number of cells, N.
Call GO SUB 62 to plot.
Don't forget to define all variables in your main program.

Martin Oakes, Freeport, IL

```
  1Ø PRINT "CELLS?"
  12 INPUT N
  14 IF N>2Ø THEN LET N=2Ø
  15 PRINT "LOOPS"
  16 INPUT L
  17 CLS
  2Ø DIM A(2Ø)
  3Ø FOR I=1 TO L
  4Ø LET J=RND(N)
  5Ø LET A(J)=A(J)+1
  6Ø NEXT I
  62 LET S=1
  63 FOR I=1 TO N
  64 IF A(I)<24 THEN GO TO 66
  65 LET S=A(I)/24+1
  66 NEXT I
  7Ø FOR I=1 TO N
  75 IF I<1Ø THEN PRINT " ";
  8Ø PRINT I; "="; A(I)
  9Ø LET M=A(I)/S
 1ØØ IF M>24 THEN LET M=24
 11Ø FOR K=1 TO M
 12Ø PRINT "▬";
 13Ø NEXT K
 14Ø PRINT
 15Ø NEXT I
 155 PRINT "NUM=";N,"LOOPS=";L,"
SCALE=";S
 16Ø STOP
SYNTACTIC SUM= 29619
```

LINE RENUMBERING

This program renumbers the line numbers in any other program in memory. It cannot renumber GO TO and GO SUB statements, but can change line numbers to increment by any step you choose, which is handy if you stick lines in while writing your program so that you don't get even steps--1Ø, 2Ø, etc.

Most users will want to skip the REM statements and most or all of the PRINT statements. I use a 14-line version. Load this routine before writing a new program, then use it to renumber the lines, adjusting the GO TO and GO SUB statements. It makes a neater product. Don't forget to include a STOP in your program. This program will not work on programs loaded from tape because loading erases everything in RAM.

You can renumber any program in memory, except this one, if you know the address of the first line number. Let S=1 less than that address. Since the first user program always starts at 16424, I started S at 16423.

Martin Irons, Goshen, NY

```
9ØØØ REM RENUMBER FIRST PROGRAM.
 PROGRAM MUST END WITH STOP. REN
UMBER GO TO/GO SUB YOURSELF. DO
NOT ATTEMPT TO RENUMBER THIS PRO
GRAM.
9Ø1Ø LET S=16423
9Ø2Ø PRINT "START WITH FIRST LIN
E AT?",
9Ø3Ø INPUT N
9Ø4Ø PRINT N
9Ø5Ø LET F=N
9Ø6Ø PRINT
9Ø7Ø PRINT "RENUMBER BY?",,
9Ø8Ø INPUT D
9Ø9Ø PRINT D
91ØØ GO SUB 92ØØ
911Ø LET S=S+1
912Ø IF PEEK(S)=118 THEN GO SUB
92ØØ
913Ø IF PEEK(S)=248 THEN GO SUB
93ØØ
914Ø GO TO 911Ø
92ØØ REM SUBROUTINE TO POKE NEW
LINE NUMBER
921Ø POKE (S+1),(N/256)
922Ø POKE (S+2),(N-(N/256)*256)
923Ø LET N=N+D
924Ø RETURN
93ØØ REM END OF PROGRAM
931Ø PRINT
932Ø PRINT "FIRST PROGRAM WAS"
933Ø PRINT " RENUMBERED FROM ";F
;" BY ";D
934Ø PRINT
935Ø PRINT "LAST LINE NUMBER= ",
(N-D)
9399 STOP
SYNTACTIC SUM= -27966
```

ANNOTATED 4K ROM LISTING

This is the 4K ROM SAVE routine listing, including the original designer's notes. Because of minor differences in assemblers, we used E8LINE to stand for E-LINE and < for SHL. We verified this assembly against the ROM using Memory Window (Jan.81 p.6 & Feb.81 p.2).

As you see, the processor does all timing—the 5 sec. quiet leader as well as the 9 pulses for a 1 and 4 for Ø.

To change the memory area SAVEd, you must at least POKE E-LINE at 16394 (4ØØAH) prior to calling this routine.

You can also create your own 5 sec. silence, LD HL with a hex starting address, then transfer control to Ø1CBH.

The LOAD module follows SAVE starting at Ø2Ø6H and ending at Ø255H. E-LINE appears at Ø21AH and Ø251H; LD HL, 4ØØØH at Ø21ØH.

```
E8LINE 400A 00105   00198
ENDBYT 01F8 00115   00191
LS8    0203 00204   00161 00190
LSAV2  01BA 00158   00166
LSAV27 01CB 00168   00192
LSAV3  01CE 00169   00188
LSAV4  01D6 00174   00182
LSAV5  01E8 00184   00185
NEWLIN 0283 00110   00204
```

```
01B6            00100           ORG     01B6H
400A            00105 E8LINE:   EQU     400AH
0283            00110 NEWLIN:   EQU     0283H
01F8            00115 ENDBYT:   EQU     01F8H
                00155 ;SAVE:                            ;SAVE FROM 4000H
01B6 D1         00156           POP     DE              ;TO (E-LINE)
01B7 11CB12     00157           LD      DE,4811         ;LEAD-IN
01BA 3E7F       00158 LSAV2:    LD      A,7FH
01BC DBFE       00159           IN      A,(0FEH)        ;SET OUTPUT = ZERO &
01BE 1F         00160           RRA                     ;READ BREAK KEY
01BF 3042       00161           JR      NC,LS8          ;IF BREAK PRESSED
01C1 10FE       00162           DJNZ    $               ;WAIT 256*13+63 T-
01C3 1B         00163           DEC     DE              ;STATES (1.04 MS)
01C4 7A         00164           LD      A,D             ;EACH TIME ROUND
01C5 B3         00165           OR      E               ;5 SECS IN ALL
01C6 20F2       00166           JR      NZ,LSAV2
01C8 210040     00167           LD      HL,4000H        ;NOW DUMP RAM
01CB 1108F8     00168 LSAV27:   LD      DE,248<8+8
01CE CB06       00169 LSAV3:    RLC     (HL)
01D0 9F         00170           SBC     A,A
01D1 E605       00171           AND     5               ;5 IF A 1, 0 IF 0
01D3 C604       00172           ADD     A,4
01D5 4F         00173           LD      C,A             ;E=9 IF 1, 4 IF 0
01D6 D3FF       00174 LSAV4:    OUT     (0FFH),A        ;TONE BURST
01D8 0624       00175           LD      B,36
01DA 10FE       00176           DJNZ    $
01DC 3E7F       00177           LD      A,07FH
01DE DBFE       00178           IN      A,(0FEH)        ;SET O/P TO ZERO &
01E0 0623       00179           LD      B,35            ;READ BREAK KEY
01E2 10FE       00180           DJNZ    $
01E4 0D         00181           DEC     C
01E5 20EF       00182           JR      NZ,LSAV4
01E7 42         00183           LD      B,D
01E8 00         00184 LSAV5:    NOP                     ;INTER BIT GAP
01E9 10FD       00185           DJNZ    LSAV5
01EB 16FE       00186           LD      D,254
01ED 1D         00187           DEC     E
01EE 20DE       00188           JR      NZ,LSAV3
01F0 1F         00189           RRA
01F1 3010       00190           JR      NC,LS8          ;IF BREAK PRESSED
01F3 CDF801     00191           CALL    ENDBYT
01F6 18D3       00192           JR      LSAV27
                00193
                00194 ;ENDBYT:                          ;COMMON CODE FOR
                00195                                   ;LOAD & SAVE AT
01F8 23         00196           INC     HL              ;END OF BYTE.
01F9 EB         00197           EX      DE,HL           ;72 T-STATES INCL
01FA 2A0A40     00198           LD      HL,(E8LINE)     ;CALL & RET IF RET
01FD 37         00199           SCF
01FE ED52       00200           SBC     HL,DE
0200 EB         00201           EX      DE,HL
0201 D0         00202           RET     NC              ;UNLESS END  DUMP
0202 E1         00203           POP     HL              ;DISCARD RET ADDR
0203 C38302     00204 LS8:      JP      NEWLIN          ;FROM ENDBYT
```

SOFTWARE REVIEW

SUPER ZX80 INVASION
Price: $14.95+$1.50 SH
RAM reqd: 1K
ROM reqd: 4K
Type of program: Game
Printed listing? No
Program listable? No
Screen prompts? Few
Easy to load? No
Challenge: Moderate
Display: Excellent
From: Softsync, Inc., P.O. Box 480, Murray Hill Station, New York, NY, 01058.

This game creates impressive continuous graphics on the ZX80, but limited memory imposes severe limitations on convenient play.

Softsync's package includes a one-page guide to loading and playing the game in addition to a cassette containing programs for both 1K ZX80s and 2K MicroAces. We reviewed only the 1K version.

As in Space Invaders, the object is to shoot down a field of space creatures before they get you. In this 1K version, you have no fortresses to hide behind: it's just you vs. 12 spacethings.

Although Softsync's guide provides excellent instructions for loading ZX80 invasion into 1K, we had a lot of trouble. Turn up your recorder's volume to maximum at first and decrease each time you try. You'll know you've succeeded when a 1ØØ REM statement appears on the screen.

Super ZX80 Invasion has little error-handling capability. Typing RUN instead of GO TO 1 to start play destroys the game's programming. If you do not input 1, 2 or 3 when choosing a level of play, the computer will crash and you'll have to reload.

Play is inconvenient. You cannot change the game's level of difficulty without turning off the ZX80 and reloading the program. In order to conserve memory, the author did not provide for running scores on the display. The only indication you have of how you are doing is a small number on your ship telling how many hits remain before you lose. The time between rounds is so short that I often couldn't tell who'd won.

Despite its limitations, ZX80 Invasion is a huge accomplishment. Never before has anyone been able to show continuous graphics in 1K memory space. The game is a moderate challenge, because you have nothing to hide behind and the spacethings fire quickly as soon as you appear before them.

ZX80 Invasion uses machine-code instructions in a REM statement, making it unlistable. Deleting the REM line displays the rest of the short basic program. ZX80 Invasion's author won't tell us how he accomplished the continuous display, but we're working on it.—SB

VIDEO DISPLAY NOTES

(Ed. note: This article describes the gut workings of the MicroAce/ ZX80 video display and is not intended for beginners. Jim has more extensive notes; check his classified ad on p.16.)

The CPU-driven video in the MicroAce/ZX80 is an incredible mix of software and hardware. the central software is the ROM "idle" routine between 13CH and 1ACH. This subroutine drives the video display, controls vertical sync pulses, and reads the keyboard during those pulses. This information was inferred from the schematic and a disassembled ROM listing (not from the designer) and gives an overall idea of the workings of the video display. Hardware names like U10 (IC9) mean MicroAce (ZX80) components.

The idle routine updates the frame counter used by RND, then checks the keyboard, shifting a Ø through the B register to read all the keyboard half-rows. This routine exists at 188H if a keypress is detected. When it's ready to display a frame of video, it jumps to a nonexistent part of memory, 32K bytes higher than the real display file. The CPU tries to read from this RAM display file, but A15 is high, the routine is not HALTed and is in an M1 (instruction fetch) cycle, and D6' is not high, so U20 and 23 (IC16 and 17) allow the open collector buffers U15 and 16 (IC14 and 15) to pull the CPU (unprimed) side of the data bus low. So the CPU sees only zeros— machine code for NOP. Meanwhile, the actual RAM character is stored in an 8-bit latch U6 (IC5), whose inputs are the primed side of the data bus. During the refresh time, that byte is part of a ROM address to get the bit pattern for one of the 8 horizontal lines making up the screen character (stored in the character generator starting at ØEØØH). The bit pattern leaves the ROM on the primed data bus, goes through resistors R5-12 (R4-11) and latches into video shift register U1Ø (IC9). This bit pattern, clocked out at 6.5 MHz, is video.

This process continues as the CPU tries to read consecutive instructions. But eventually, a NEWLINE (76H) appears in the display file. Because NEWLINE has bit 6 high, the open collector NOP writer is not enabled, so the CPU reads 76H as an instruction. In Z80 code, 76H is a HALT command.

The only way out of a Z80 HALT state is an interrupt or a reset. The interrupt line is sampled during refresh time and is connected to address line A6. When the refresh counter in the CPU (preset in the idle routine) comes to an address with A6 low, it generates the interrupt that signals the end of a scan line. One result is a special I/O read cycle, kicking a ripple-through one-shot that generates the horizontal sync pulse and increments the 3-bit counter U14 (IC21) that keeps track of which scan line the ROM gives the bit pattern for. The other result is a CALL to an interrupt handler at ØØ38H. This routine decrements C, the scans per line counter, and if necessary, B, the lines per display counter. It returns to non-existent memory to either re-read the same characters for another scan, or to read the next line of characters. When done, it returns to the idle routine.

The IN instruction that reads the keyboard also sets the sync line low for the vertical sync pulse. Later, after a carefully timed instruction sequence, an OUT instruction at 196H ends the vertical sync pulse. Of course, the idle routine returns to BASIC when a keypress is detected.

Jim Williams, Calumet City, IL

NEW 8K ROM FEATURES

Sinclair's 8K ROM may well be a reality to North American owners by now (see story p.1). Here is an updated list of the redesigned 8K ROM features not available on the 4K ROM and a brief explanation of their functions. Check the 8K manual for additional explanations. Note the demise of the READ-DATA and DRAW-UNDRAW statements since our last report of ROM features. The printer commands and some ZX81-specific statements, such as FAST and SLOW, are new.

FEATURE	FUNCTION
COPY	outputs screen display to printer
FOR..STEP	increments FOR-TO statements by a definable number
K-L MODE	changes cursor mode between keyword and letter
LLIST	lists to printer instead of screen
LOAD F	loads program named F and its variables from cape
LOAD ""	loads the first program found on tape
LPRINT	directs output to printer instead of acreen
PAUSE N	sends display file to screen for N frames or until any key is pressed
PLOT M,N	sends the plot position (a system variable) to (M,N), line M and column N, and blacks in that pixel
PRINT AT M,N	moves the print position to line M, character N
SAVE F	saves program and variables on tape, naming it F
SCROLL	scrolls display file up 1 line, losing top line and making space at bottom
UNPLOT M,N	like plot, but blanks pixels instead of blackening them
FAST, SLOW	selects display mode, useful only for ZX81 (not currently available in North America)

In addition, the new ROM provides these functions:

ARCOS	arc cosine in radians
ARCSIN	arc sine in radians
ARCTAN	arc tangent in radians
COS	cosine in radians
EXP	e^x
INKEY$	reads the keyboard, giving character representing key pressed, otherwise empty string
LEN X$	number of characters in string X, including spaces
LN	natural log
PI	3.1415927
SGN	yields -1, Ø, +1
SIN	sine in radians
SQRT	square root
TAB X	moves print position to column X
TAN	tangent in radians
VAL	(followed by a string) evaluates X as a numerical expression, X must not contain quote character

All graphics can be entered directly from the keyboard; SHIFT 9 enters graphics mode until terminated by (NL) or another SHIFT 9.

CAI INSTRUMENTS Introduces the 'WIDGET'® Series of Peripherals for the ZX80

The 'WIDGET'® Series of Peripherals open the door to real computing.

'WIDGET'® $49.95

The Widget interface board allows your ZX80 to communicate with the rest of the world through both a serial RS/232 data bank and a parallel TTL input and output parts.

'WIDGET'® $99.95 Widgitape

The Widgitape is a revolutionary tape storage system that provides the features of high priced floppy disks at a fraction of the cost.

'WIDGET'®$99.95 Widgiprint

The Widgiprinter is a 28 column thermo printer that provides a hard copy of results from your programs at a price you can afford.

Have you developed a unique option or application for your ZX80? We at CAI would be interested in discussing the possibility of marketing your idea for you. Contact us.

Remember this new "WIDGET" Series can open the door to real computing for your ZX80.
Send Check, Money Order, or Credit Card Company to:

CAI Instruments, Inc.
2559 Arbutus Court
Midland, MI 48640

Or order by phone (517) 835-6145 giving your credit card number.

Quantity	Description	Unit Price	Total
	WIDGET		
	WIDGIPRINT		
	WIDGITAPE		
	TOTAL		

Card No. Visa _____
Mastercharge _____
Exp. Date _____
Name _____
Address _____
City _____ State _____ ZIP _____
Michigan residents add 4% sales tax

DEAR EDITOR

How can I adapt the US ZX8Ø to use it in worldwide travel?

J. Atteslander, Agincourt, Ontario

Remove Diode D11 (from data line 6 to KBD). This changes the frame rate to accomodate European TV systems. Tune the TV to channel 3 (equivalent to US channel 2) and use 22ØV AC to 9V DC power supply. If you use your US power supply, your voltage output may burn out the transformer.—AZ

As a beginner to computing, I have trouble running my own programs because of errors. I realized that if I could adapt parts of programs in SYNTAX, I could learn the language faster. Can you publish a flowchart or some indication of the function of the important parts of programs? For example, in Cryptoquote (Nov. 8Ø), 1 section does the frequency count and another substitutes letters. Which lines are essential for each?

Richard Lunberg, Santa Clara, CA

The best way for beginners to learn to program is to just do it, using good examples from others as a guide. We will try to provide more explanation about how our published programs work, although space limitations will preclude flowcharts. Please write anytime you want further explanation of programs in SYNTAX. We try to answer all letters, but those asking specific questions are more likely to get answered quickly and thoroughly. Cryptoquote is not a good program for a beginner to try to unravel. It does a neat job of putting lots into little space by using only 1 array. Lines 4ØØ-54Ø calculate and generate the frequency table; line 51Ø prints this. To find out what parts of a program do what, try changing pieces and examining the result. One good thing to try is changing a variable to a constant and seeing where it is printed.—AZ

I would like to see a complete listing of 8K ROM BASIC addresses, entries, subroutines, and tables published soon. Using Linsac's ZX8Ø Companion, I wrote some machine level programs that I want to use without waiting for another Linsac book. Also I have written repeatedly to CAI Instruments for information on their Widget/tape/printer to no avail. How about an in-depth product review? In general, your product reviews leave much to be desired.

Robert B. Keller, Rochester, MN

Sinclair Research has no plans to release its 8K ROM listing soon. We have, however, a complete annotated listing of the 4K ROM that will soon be available to SYNTAX readers. Bob Swann of CAI said he has spoken with you by phone. We will certainly provide an in-depth review of CAI's printer, tape drive and Widget as soon as we receive ours, but we cannot judge equipment we haven't seen. When we make new product announcements, we give you all the information we have as we go to press. These are not intended to be reviews, just information.—AZ

I use an Olympus micro-cassette Pearlcorder SD and have never had a bit of trouble, using either AC or batteries. The small size complements the ZX8Ø nicely.

Robert W. Golsmith, Wyckoff, NJ

These readers would like to hear from others in their areas. If you would like to contact local ZX8Ø/MicroAce users, send us your name, address and phone number and we'll print them when space

permits.—AZ

Gene Hammond, P.O. Box 505,
Carmel, CA, 93921, 408/624-7737

David Foyt and Lee Tharrett, 3906
Wilbert Rd., Austin, TX, 78751

INTRO TO MACHINE LANGUAGE

This starts our series on machine language programming for beginners. You'll need to review our previous colums on binary and hex number systems and see the story on addresses this issue.

Last month we published the routine to check how much ZX80 memory is connected, courtesy of Sinclair Research:

```
1 POKE 17000,33
2 POKE 17001,11
3 POKE 17002,0
4 POKE 17003,57
5 POKE 17004,201
6 PRINT USR (17000)-16383
```

—A machine language program in disguise.

Some of these BASIC commands may be unfamiliar to you. POKE places decimal values between 0 and 225 in specified addresses (see p.15 about addresses). USR begins at the address specified and executes a machine language subroutine until it reaches a code (201 or C9 hex) that tells it to return to the point in the BASIC program from which it left, much like a subroutine's GO SUB and RETURN commands.

Machine language controls the computer by manipulating numbers in registers and memory. A register is the computer's scratchpad for holding and calculating numbers, usually between 0-225. Memory is used to store the numbers, which can represent the numbers themselves, characters in the ZX80 character code, or opcodes (machine language commands). Opcodes are like computer shorthand.

The BASIC system uses some RAM space, called the stack, to evaluate expressions (like A+B/C), save the original value of changing variables (as in a FOR-NEXT statement) and execute subroutines. Register SP, the stack pointer, tells the computer where the stack ends and spare memory begins. The plan behind this program is to check where the stack pointer is and add back the number of bytes in the stack. This will give the total bytes of RAM available.

Let's look at the values that this BASIC routine pokes into memory:

```
33  11   0  57  201  (decimal)
21  0B  00  39   C9  (hex)
```

Opcode 33 tells the computer to load the next two bytes, 11 and 0, into a working register pair, HL.

The integer loaded into HL is 11 decimal, or 000B hex. 11 and 0 represent only one integer value. The ZX80 stores all integers in two bytes. The programmer knew by testing a working ZX80 that the stack has 11 bytes in it when this program executes. Because the stack pointer is at the top of memory and moves down as the stack above fills, it will be 11 bytes from the top of memory. To load HL, you need to use 2 bytes (remember, HL is a register pair), so you add 11 then 0 to compensate for the 11 bytes in the stack. The first byte loaded is called the lower byte, the second is called the higher byte. "Lower" and "higher" refer to the byte's position in the register, not to the magnitude of the number loaded.

Opcode 57 tells the Z80 to add the contents of HL (000B hex) to the value of SP and places the answer back in HL. So HL's new value is that of SP plus 11.

Opcode 201 is the return to BASIC command.

These codes are processor-specific, that is, all Z80 processors (like the ones in ZX80s

and MicroAces) use the same codes, but other processors like the 65Ø2 or 68ØØ do not. You can find these codes in books on machine language programming or code listings. Two such books are Rodnay Zaks' Programming the Z8Ø, published by Sybex, and the Z8Ø-Assembly Language Programming Manual from Zilog (who first developed the Z8Ø processor). See p.12 for information on ordering the Zilog manual.

The computer cannot distinguish opcodes from data except in context. That is, it must be told where to begin reading, or it will execute the wrong instructions (if it can read the codes at all). The BASIC USR command sends the computer to the correct address to start reading so the machine language codes will make sense.

In the routine above, the ZX8Ø pokes five integers into addresses starting at 17ØØØ. USR(17ØØØ) tells it to begin reading at address 17ØØØ and execute machine language instructions until it hits the return at 17ØØ4. Then the computer pops back into BASIC and evaluates USR(17ØØØ)-16383. USR(17ØØØ) evaluates to the number in HL when the computer returns to BASIC. Register HL now contains the sum of SP and 11, or 174Ø7, the highest consecutive RAM address. Subtracting 16383 removes the possible ROM locations, giving the answer 1Ø24 RAM bytes for a 1K machine.

MACHINE CODE MONITOR

This utility program enables you to edit and execute hex Z8Ø machine code. It derives from a much simpler one in 3Ø Programs for the ZX80, published by Melbourne House Ltd. It can help you learn to program the Z8Ø in its native language.

If you're new to machine language, try this program with the Syntactic Sum assembly listing (p.8 Feb. 81). The actual listing is the single column after 43EØ (the first column lists hex addresses, the third, fourth & fifth show human-readable mnemonics for the instructions, & the sixth explains what the instructions do). Input the hex numbers in the listing, 2 digits at a time. Thus, respond to the number of locations you need with the number of 2-digit hex numbers to enter (31 in this case). Then type in the hex numbers, 2 digits at a time, hitting (NL) after each pair. Type 21 in response to the first data request, 28 to the second, 4Ø to the third, ED to the fourth, and so on. After you enter EC, the last number in the listing, the computer still expects a number, so hit (NL) to enter ØØ. (This is because line 12Ø of the BASIC monitor program sets up a FOR-NEXT loop from Ø to L, L being the number of locations you needed. The number of locations from Ø to L equals L+1, so the computer expects L+1 entries.)

When the BASIC program gets to line 27Ø, it executes a machine language call, which means it runs the machine language program you typed in. Because it is the Syntatic Sum program, the answer you get should be the Syntactic Sum of the hex monitor program.

Always respond to line 6Ø in decimal; use hex for all other input. The result of the USR subroutine call is decimal.

You can reserve between 9 and 1ØØ locations. If you need fewer than 9, just hit (NL) to enter ØØ (for NOP, or no op code) to fill the remaining addresses. After writing to all locations, the computer asks you for any changes. Type Y to change the data. The program re-accesses each location and you can enter new data or hit (NL) to keep the original data. Any answer other than Y executes

The machine code, prints the result and asks STOP? Respond Y to stop; default re-accesses the machine code.

Besides returning control to the BASIC interpretor from the machine code, C9H (meaning C9 in hex) causes "STORAGE LOCATIONS" to be printed as a convenience header. But these locations may be used for code or storage.

Line 2ØØ avoids crashing the program when machine code listings fill the screen (see p.123 of the ZX8Ø operating manual).

This program starts listings at address 16963 and provides up to 1ØØ bytes of storage. Bytes deleted from the monitor listing (characters and tokens, not variables, numbers or dimensioned arrays) free up the same number of bytes for hex code. Note the resulting shift in the starting address of the listing. Reduce the starting address by the number of bytes you delete (2 for each line number, 1 for each keystroke & 1 for end of line).

Matthew J. Johnson, Malden, MA

```
  1Ø REM BE SURE TO LEAVE RESULT
  2Ø REM IN THE HL REGISTER AND
  3Ø REM INCLUDE A RETURN.
  4Ø PRINT "HOW MANY LOCATIONS N
EEDED?"
  5Ø PRINT "(FIGURE ONE BYTE/LOC
.)"
  6Ø PRINT ":::9<NUMBER<1ØØ:::"
  7Ø INPUT L
  8Ø DIM A(L)
  9Ø LET S=PEEK(16392)+256*PEEK(
16393)+9
```

```
 1ØØ CLS
 11Ø PRINT "LOCATION","CONTENTS"
 12Ø FOR I=Ø TO L
 13Ø PRINT S+I,,
 14Ø INPUT A$
 15Ø IF A$="" THEN GO TO 18Ø
 16Ø LET V=16*CODE(A$)+CODE(TL$(
A$))-476
 17Ø POKE S+I,V
 18Ø LET V=PEEK(S+I)
 19Ø PRINT CHR$(V/16+28);CHR$((V
 AND 15)+28)
 2ØØ IF PEEK(16421)=9 THEN CLS
 21Ø IF V=2Ø1 THEN PRINT "STORAG
E LOCATIONS:"
 22Ø NEXT I
 23Ø PRINT
 24Ø PRINT "ANY CHANGES?"
 25Ø INPUT A$
 26Ø IF A$="Y" THEN GO TO 1ØØ
 27Ø PRINT "RESULT OF USR IS: ";
USR(S)
 28Ø PRINT "STOP?"
 29Ø INPUT A$
 3ØØ IF A$="Y" THEN STOP
 31Ø GO TO 1ØØ
SYNTACTIC SUM= -23248
```

BEGINNER'S ROM AND RAM ADDRESSES

In the last several issues we learned different computer number systems. The ability to use these systems is of more than just mathematical interest. This month we'll talk about ROM and RAM addresses in 4K ROM, 1K RAM machines and later learn how to access these addresses using hex and binary numbers.

ROMs and RAMs are both electronic memory devices. They think only in terms of on-off, or true-false, or 1-Ø. Everything is black and white to a computer. The difference between the 2 is that a ROM is read only, meaning that it is programmed by the manufacturer and you cannot change that program, while a RAM is random access, meaning that you can input programs or data, and you can change them. As you've seen, a RAM program you've typed in is lost when you disconnect the power. The ROM program is permanent and comes back every time you turn on the computer.

Everything in a ROM or RAM is stored in specific locations, or addresses. Each address holds 1 byte (or 8 binary digits, called bits). Think of ROMs and RAMs as long rows of mailboxes. Each box has a unique address. When you look at the contents of ROM or RAM, you see copies of the mail that has been placed in a given box and each box holds only 1 piece of mail. You often need to see the contents of 2 or more boxes to understand the message because each message must be understood in context. Looking at the mail does not remove it; after you're done the mail is still there because the computer showed you only a copy.

A 4K ROM has 4Ø96 bytes of memory, so it has 4Ø96 addresses where those bytes reside. ROM addresses start at Ø and go to 4Ø95. A 1K RAM, as we've seen before, has 1Ø24 bytes of memory, so it also has 1024 possible addresses (Ø-1023) for those bytes to be in. 1K RAM addresses start at 16384 and go to 174Ø7 (16384+ 1Ø23). Why don't RAM addresses start at 4Ø96?

The Z8ØA processor, the central brain in a MicroAce or ZX8Ø, can address 65536 (or 256x 256) locations. Each address consists of 2 bytes, or 16 bits. The ZX80 needs 1 bit as a sort of zip code, to tell which memory the address goes to, leaving only 7 bits for the rest of the address in that byte. The largest number you can represent with 7 bits is 128. The other byte still has 8 bits, or up to 256. The total number of addresses you can create is therefore 128x256, or 32768. Thus the available addresses are reduced by ½. The machine designer, in his infinite wisdom, defined the ROM space as addresses Ø to 16383, containing 16384

addresses. Thus, if you wanted to add more ROM chips to your computer, you have address space available. The RAM space, then, starts at 16384, the next address in line, and theoretically goes to 32768, or 32K (2^{-1}, or 1/2 times 65536). Since you only use 1024 of these RAM addresses in a 1K machine, your actual addresses go to 17407.

Next month we'll see how to use PEEK and POKE to see and alter the contents of addresses.

CLASSIFIEDS

For sale: ZX80, hard carrying case, software, SYNTAX back issues, all for $150. J. Weiner, 5510 Pimlico, Baltimore, MD 21209 (301)367-4170.

Programs-games & utility. Also tech data, mods, plans, info, etc. Send SASE for free goodies list. P.O. Box 3073, San Jose, CA 95116.

ZX-Display: 50 page book shows how to improve displays using a unique method for reduced RAM usage. Incl listing for 6 games using method. $8 to Sum-Ware, Shawville, PA 16873

Nov/Dec SYNTAX combined reprints, $5.00, limited numbers of other back issues available, $4.00 each. Send check or credit card number to SYNTAX, RD 2 Box 457, Bolton Rd, Harvard, MA, 01451.

Video secrets revealed! This 8-page document explains in detail how ZX80 video works, including a USR routine to run the display. Also included are an explanation and USR routine to run the keyboard and some notes on ROM routines. $4.00 postpaid. Jim Williams, 262 Chappel, Calumet City, IL, 60409

Zilog Z80-CPU Z80A-CPU Technical Manual, $7.50, and Z80-Assembly Language Programming Manual, $15.00. Add 5% for postage and handling. Send check or credit card no. to SYNTAX, RD 2 Box 457 Bolton Rd, Harvard, MA, 01451.

Got something to sell or swap? Or are you looking for something special? Use SYNTAX classifieds. Reach hundreds of other readers for just $2.75 per line (4 line min.). Send your ad, typed 35 characters per line, with payment by 15th of any month for next month's issue to Classified Ads, SYNTAX, RD 2 Box 457, Bolton Road, Harvard, MA, 01451. No telephone orders, please.

SYNTAX

THE HARVARD GROUP
Bolton Road, Harvard, Mass. 01451

First Class